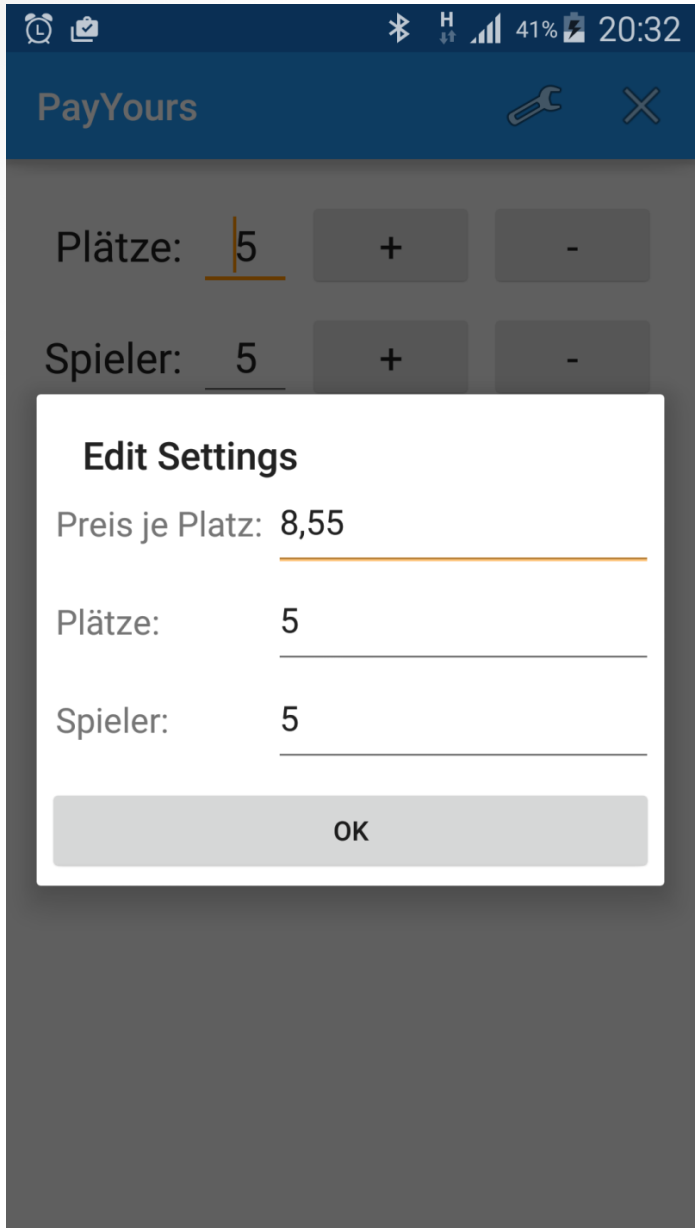
UI unter Android ➔ Activity

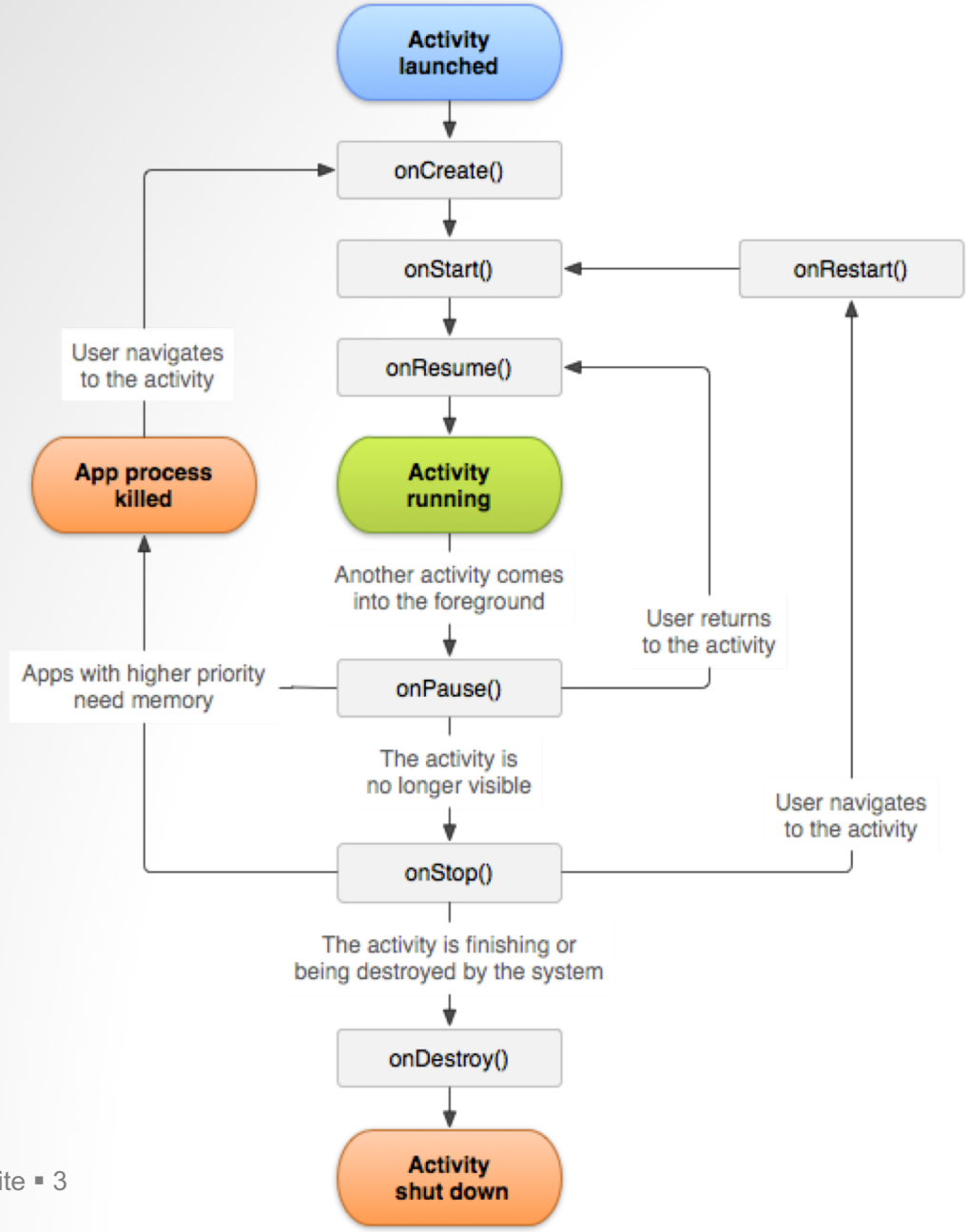


HTL LEONDING

Grundlegendes zu Activities

- Hauptzweck: Verarbeitung der Benutzerinteraktion
  - Funktion eines Controllers
- Pro Bildschirmseite wird eine Activity angelegt
- Strikte Trennung Design ⇔ Code
  - Design in res-Ordner ➔ layout
    - Automatische Anpassung an Ausrichtung, Tag/Nacht, …
    - ADT erstellt daraus statische R-Klasse
      - Ressourcencompiler *Android Asset Packaging Tool* (*aapt*)
  - Texte werden ebenfalls in Ressourcen verwaltet

Lifecycle einer Activity unter Android

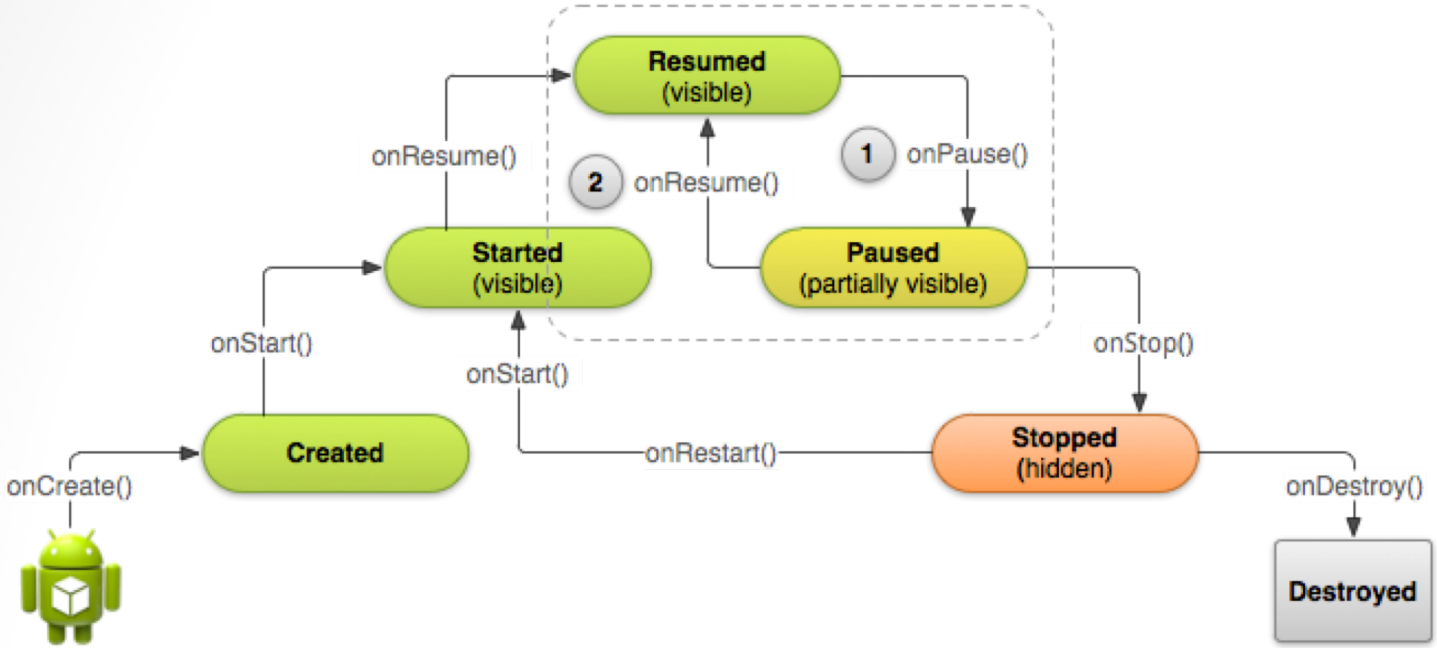


- Activity durchläuft verschiedene Zustände
- Ressourcenmangel ➔ Anwendung kann vom System geschlossen werden
- Prozess kann auch nach Destroy durch Anwender weiterlaufen ➔ schnellerer Start

HTL LEONDING

onPause() – onResume()

- Activity verliert Fokus
  - DialogActivity liegt semitransparent darüber
  - Animationen stoppen, Ressourcen freigeben
  - AndroidDeveloper



HTL LEONDING

onStop() – onStart()

- Auslöser für onStop()
  - Andere App kommt in den Vordergrund
  - In eigener App wird neue Activity aufgerufen
  - System beendet App wegen Resourcenmangels
- Aufgaben
  - Persitieren der Zustände
- onStart()/onRestart()
  - onRestart() wird nur nach onStop() aufgerufen
  - Normalerweise onStart() verwenden
  - Wiederherstellen des Zustands der Activity

onDestroy() – onCreated()

- Aufruf z.B. bei Änderung der Orientation
  - Anderes Layout ist möglich
- onSaveInstanceState() sichert alle View-Zustände (Texte, Scrollpositionen, …)

Beispiel: onSaveInstanceState()

- AndroidDeveloper

```
// some transient state for the activity instance
var gameState: String? = null
```

```
// invoked when the activity may be temporarily destroyed, save the instance state here
override fun onSaveInstanceState(outState: Bundle?) {
    outState?.run {
        putString(GAME_STATE_KEY, gameState)
        putString(TEXT_VIEW_KEY, textView.text.toString())
    }
    // call superclass to save any view hierarchy
    super.onSaveInstanceState(outState)
}
```

onRestoreInstanceState()

- Alternativ auch in onCreate()

```
override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    textView.text = savedInstanceState?.getString(TEXT_VIEW_KEY)
}
```

Beispiel CoolDroid

- Zählerstand bei Wechsel der Orientation beibehalten

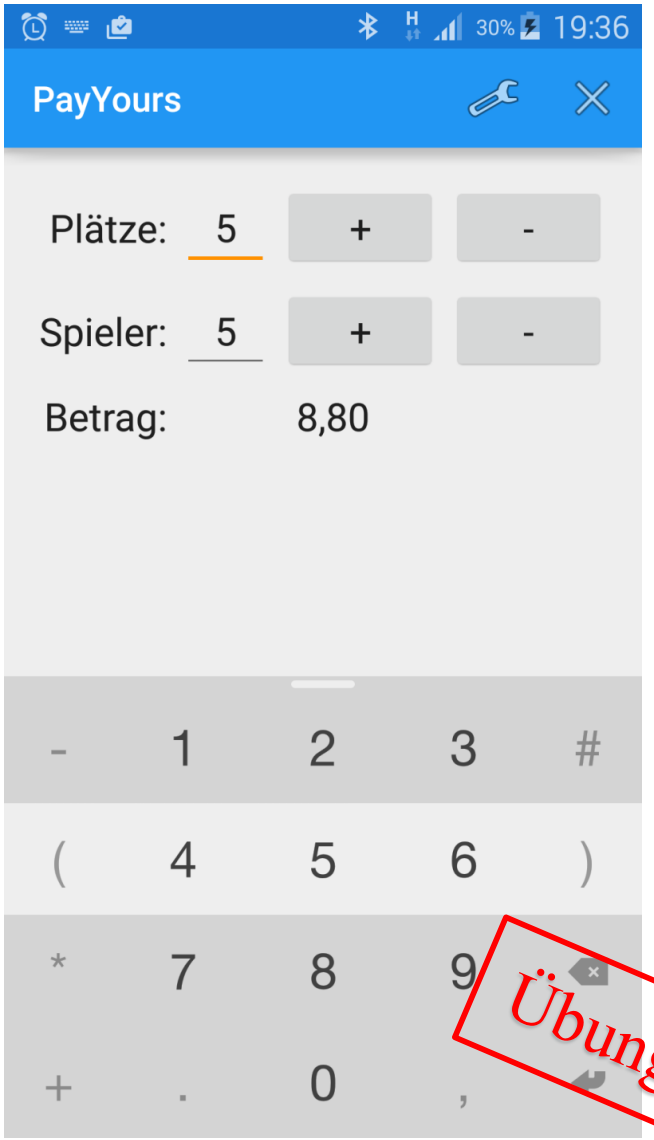
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    if (savedInstanceState != null) {
        counter = savedInstanceState.getInt(COUNTER);
        tv_message.text = getMessageText()
    }
    iv_droid.setOnClickListener {...}
}

override fun onSaveInstanceState(outState: Bundle?) {
    outState?.putInt(COUNTER, counter)
    Log.d(LOG_TAG, msg: "onSaveInstanceState(), counter: ${counter}")
    super.onSaveInstanceState(outState)
}
```

Richtlinien für Lebenszyklusmethoden

- onCreate() ➔ Objekte instanzieren, die in Activity benötigt werden (z.B. Controls)
- onResume() ➔ starten von Services und Code, der benötigt wird, wenn Activity im Vordergrund ist
- onPause() ➔ stoppen etwaiger Services
- onDestroy() ➔ freigeben von Ressourcen

Erstes einfaches Beispiel als Einstieg

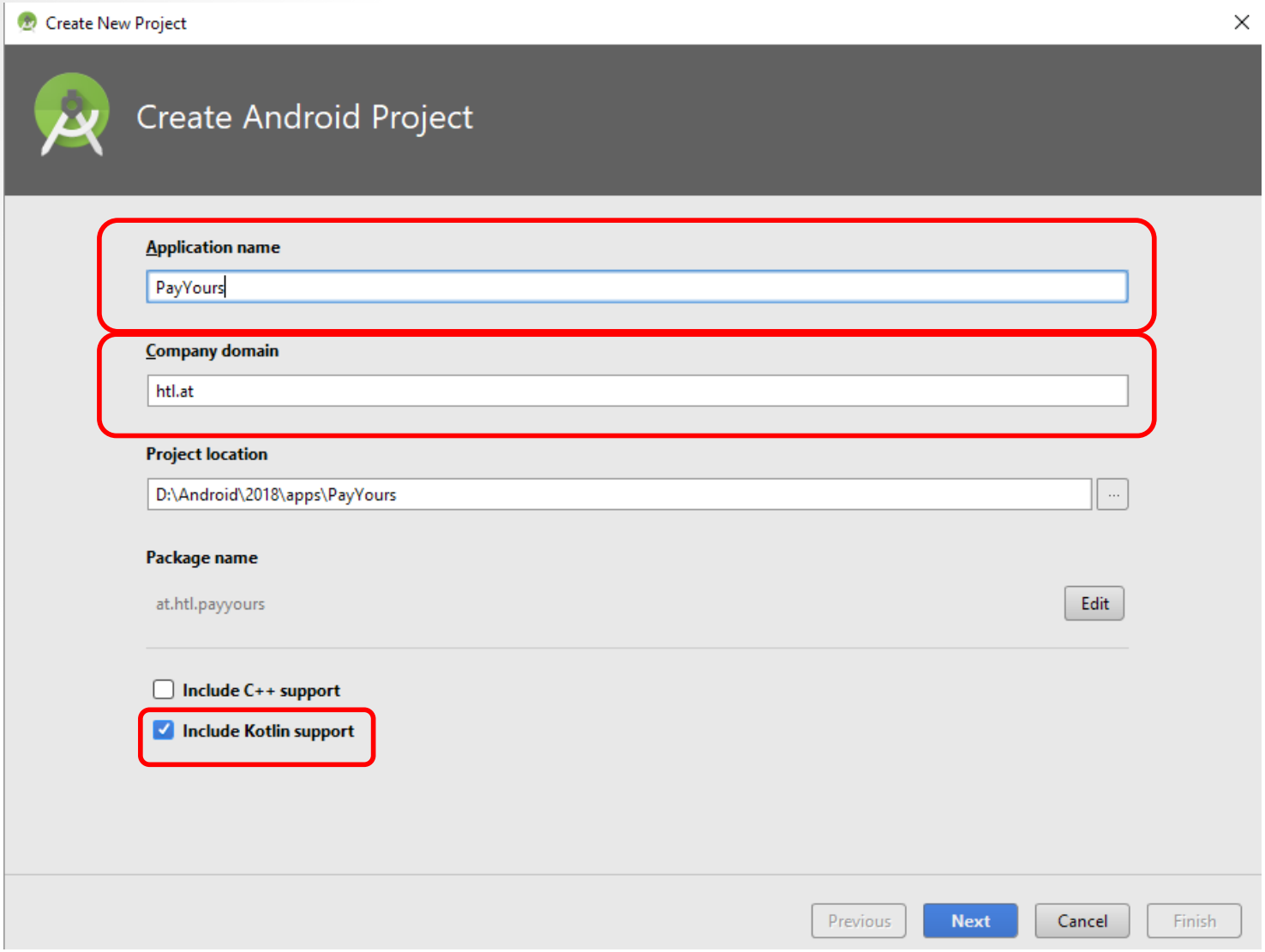
- Anforderungen
  - App unterstützt die Verrechnung von Plätzen (Tennis, Badminton, Volleyball, …) an die Mitspieler
  - Über Settings Möglichkeit, Parameter einzustellen
- Lehrziele
  - Verknüpfung von Activities über explizite Intents
  - GridLayout
  - Einfache AppBar als ActionBar



PayYours_a Anfang

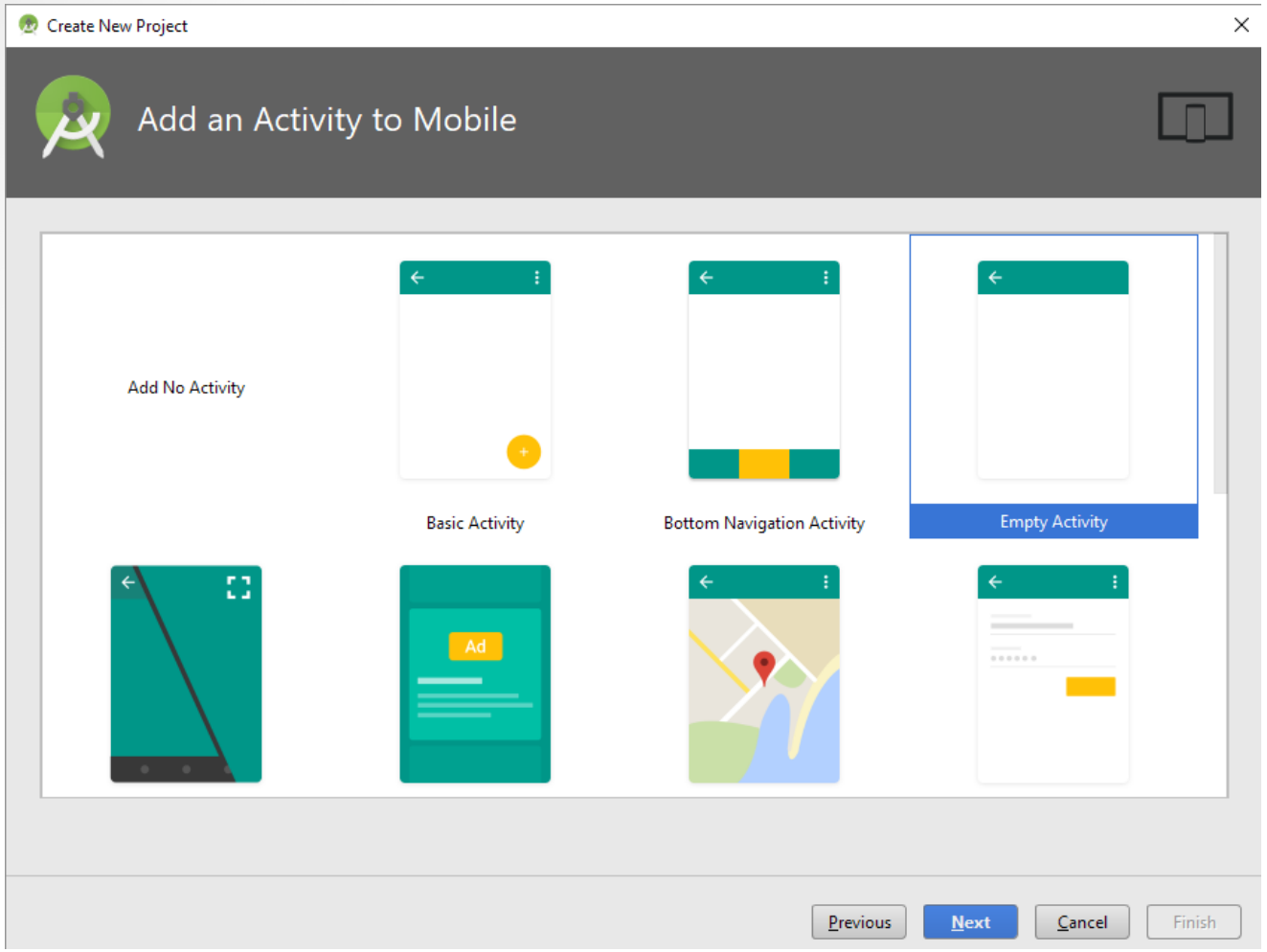
HTL LEONDING

Neues Projekt anlegen



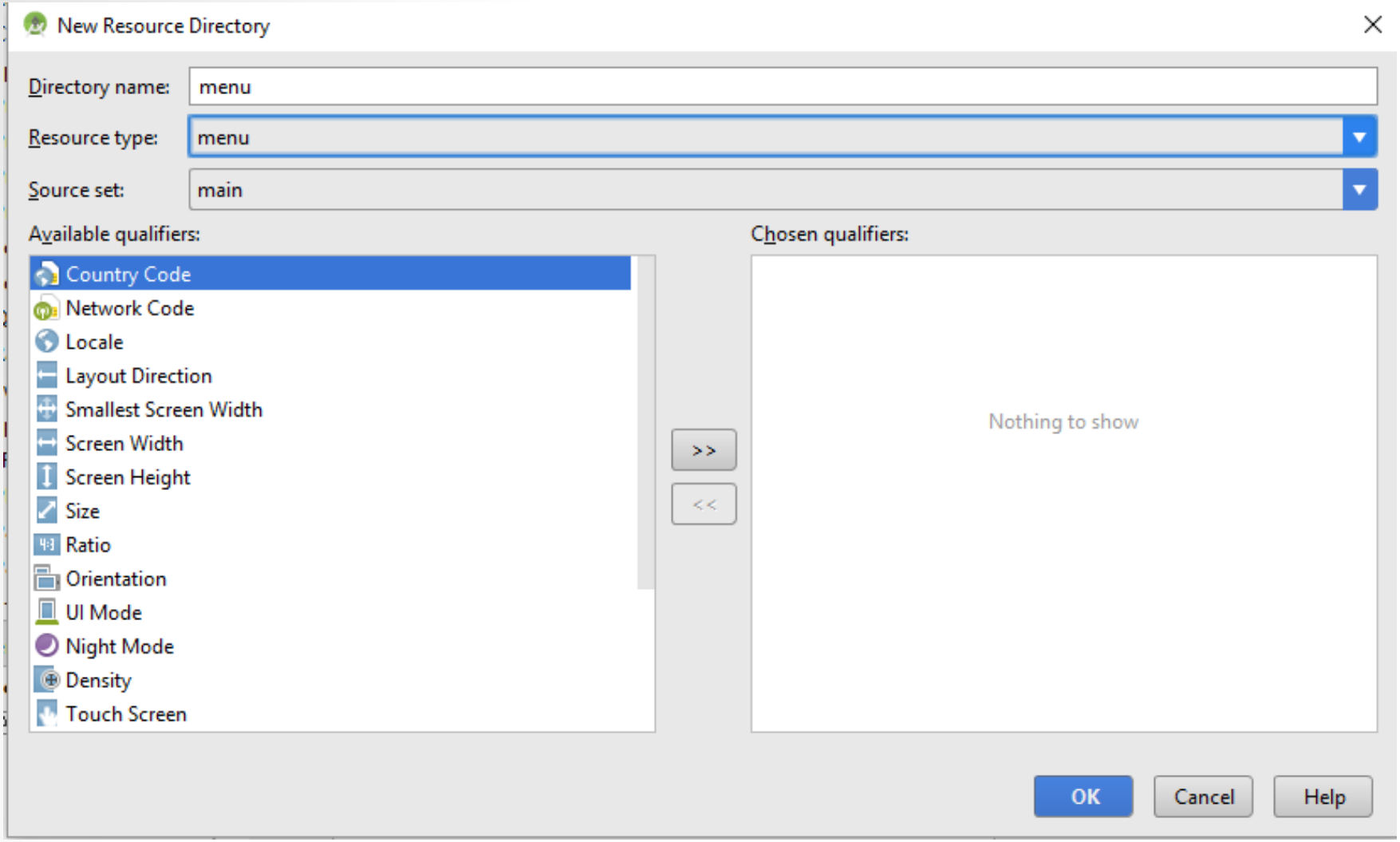
HTL LEONDING

Leere Activity



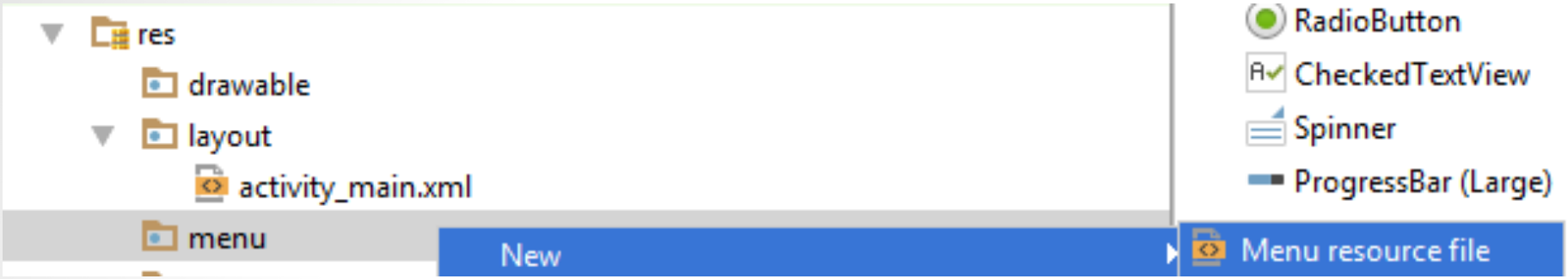
HTL LEONDING

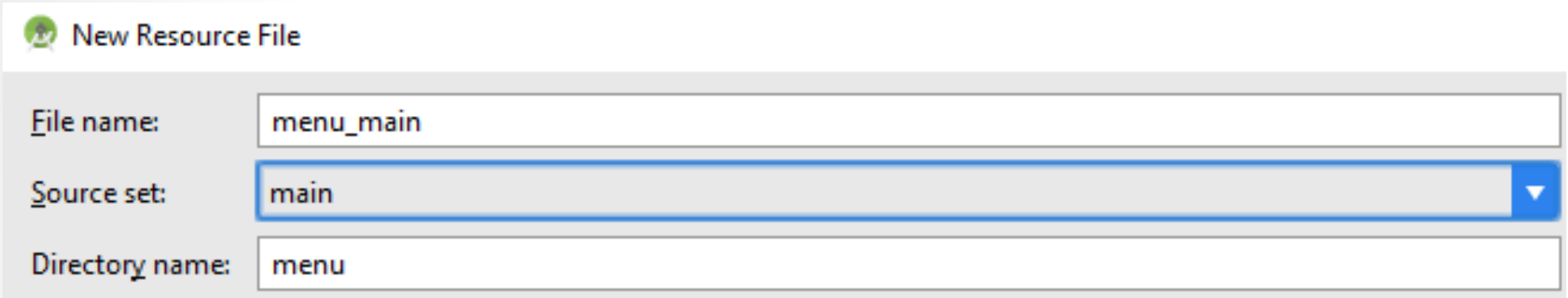
Allerdings mit OptionsMenu



HTL LEONDING

Menu-Ressourcefile anlegen





HTL LEONDING

Hauptmenü über ActionBar konfigurieren

- Menüs, die über die Menütaste des Geräts aktiviert werden
- Pro Activity existiert ein Optionsmenü
- Menüs werden als XML-Ressource (Verzeichnis `/res/menu`) oder im Java-Code definiert
- Die Definition der Menüeinträge findet in der Methode `onCreateOptionsMenu` statt
- Bei Auswahl eines Menüpunktes wird die Methode `onOptionsItemSelected` aufgerufen

Hauptmenü als ActionBar konfigurieren

- menu_main.xml
- Menüpunkte Settings und Quit in ActionBar mit Icon aus Package Android
  - Quit soll immer in ActionBar angezeigt werden

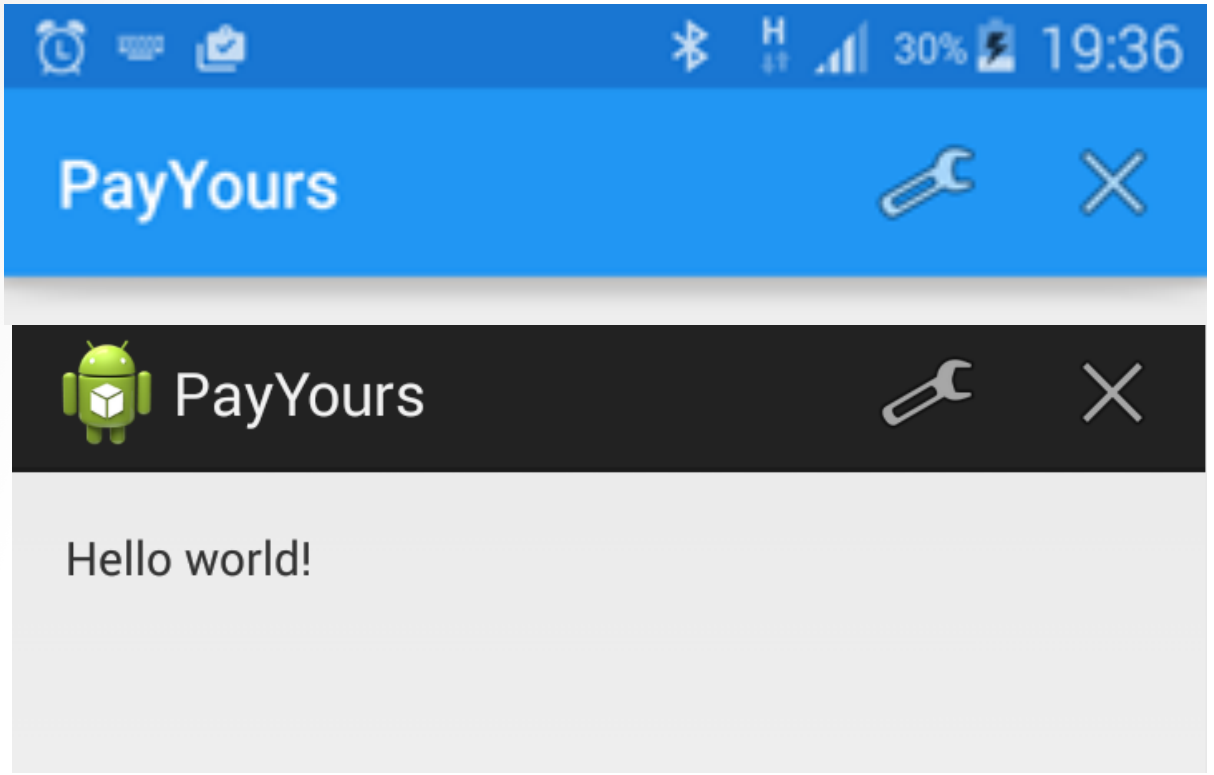
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity">
    <item
        android:id="@+id/menu_action_settings"
        app:showAsAction="always"
        android:icon="@android:drawable/ic_menu_preferences"
        android:title="Settings"/>
    <item android:id="@+id/menu_Quit"
        android:icon="@android:drawable/ic_menu_close_clear_cancel"
        app:showAsAction="always"
        android:title="Exit"/>
</menu>
```

Code für Menübehandlung in MainActivity.java

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.menu_main, menu)
    return super.onCreateOptionsMenu(menu)
}

override fun onOptionsItemSelected(item: MenuItem) = when (item.itemId) {
    R.id.menu_Quit -> {finish();  true}
    R.id.menu_action_settings -> {true}
    else -> super.onOptionsItemSelected(item)
}}
```

Ergebnis: ActionBar mit ActionButtons

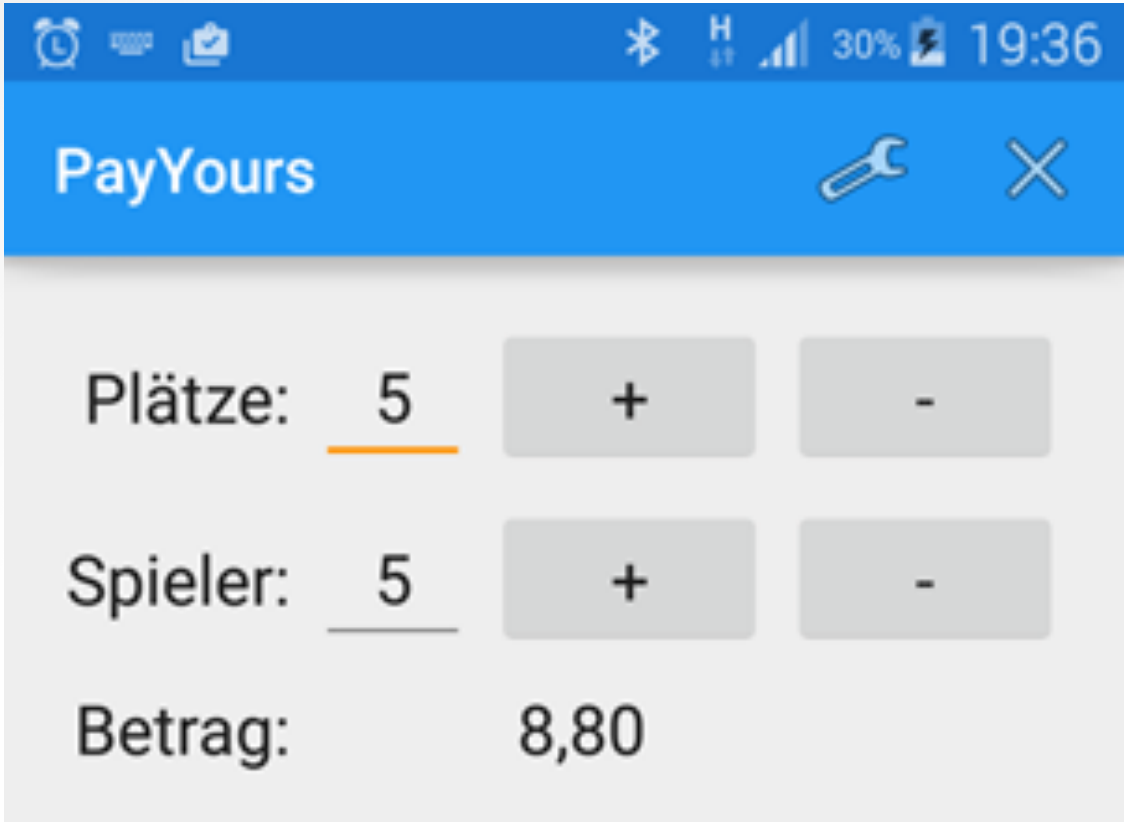
- Früher (vor Android 3/API 11) über OptionsMenu
- finish() funktioniert



PayYours_a Ende

HTL LEONDING

Gewünschtes UI für Calculator



PayYours_b Anfang

HTL LEONDING

strings.xml

```xml
<resources>
    <string name="app_name">PayYours</string>
    <string name="action_settings">Settings</string>
    <string name="activity_edit_settings_title">Edit Settings</string>
    <string name="menu_exit">Exit</string>
    <string name="txt_courts">Plätze:</string>
    <string name="txt_players">Spieler:</string>
    <string name="txt_to_pay">Betrag:</string>
    <string name="btn_plus">+</string>
    <string name="btn_minus">-</string>
    <string name="ed_courts">7</string>
    <string name="ed_players">6</string>
    <string name="txt_price_per_unit">Preis je Platz:</string>
    <string name="txt_default_courts">Plätze:</string>
    <string name="txt_default_players">Spieler:</string>
    <string name="bt_ok">OK</string>
    <string name="title_activity_settings">Settings</string>
</resources>
```

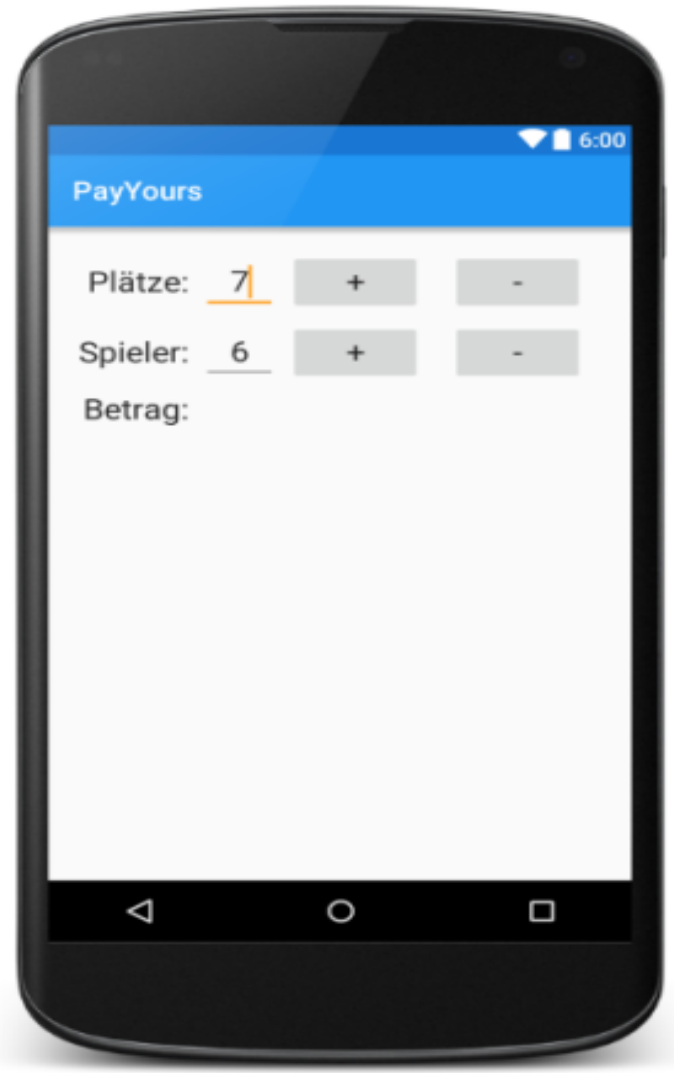
activity_main.xml ➔ GridLayout

activity_main.xml

GridLayout

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <GridLayout xmlns:android="http://schemas.and
3       xmlns:tools="http://schemas.android.com/t
4       android:paddingBottom="16dp"
5       android:paddingLeft="64dp"
6       android:paddingRight="64dp"
7       android:paddingTop="16dp"
8       android:layout_width="match_parent"
9       android:layout_height="wrap_content"
10      android:columnCount="4"
11      android:columnOrderPreserved="false"
12      android:orientation="vertical"
13      android:rowCount="3"
14      android:useDefaultMargins="true"
15      tools:context="at.htl_leonding.payyours.M
16      <TextView...>
22      <EditText...>
35      <Button...>
43      <Button...>
51      <TextView...>
58      <EditText...>
68      <Button...>
76      <Button...>
84      <TextView...>
91      <TextView...>
100 </GridLayout>
101
```

Design | Text



Abstände auslagern ➔ dimens.xml

dimens.xml

```xml
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="text_view_padding">12dp</dimen>
    <dimen name="et_numbers_width">50dp</dimen>
    <dimen name="margin_between">3dp</dimen>
    <dimen name="text_view_to_pay_width">110dp</dimen>
    <dimen name="default_padding">5dp</dimen>
    <dimen name="default_et_width">200dp</dimen>
</resources>
```

GridLayout

- Tabellenartige Darstellung
  - Aufteilung des Platzes in Zeilen, Spalten und Zellen
  - Span-Property erlaubt View, mehrere Zellen zu verwenden
  - Automatische Positionierung der Views oder Vergabe von Zeilen- und Spaltenindizes
  - Ausrichtung der Views in Zelle horizontal und vertikal möglich
  - Neuer Viewtyp Space ermöglicht Leerbelegung
- Alternative zu verschachtelten LinearLayouts und RelativeLayouts
  - Langsam und wartungsintensiv, wenn verschachtelt
- API >= 14 notwendig

TextView für Beschriftung


```
<TextView
    android:layout_row="0"
    android:layout_column="0"
    android:layout_gravity="right|center_vertical"
    android:text="@string/txt_courts"
    android:textAppearance="?android:attr/textAppearanceLarge" />
```

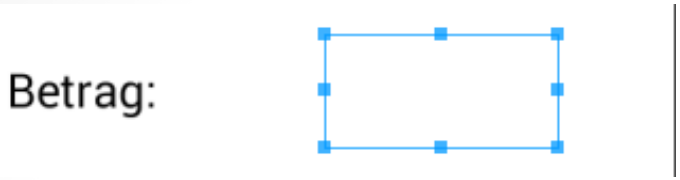
EditText für Zahleneingabe


```
<EditText
    android:id="@+id/editTextCourts"
    android:layout_width="85dp"
    android:layout_column="1"
    android:layout_gravity="center_vertical|center_horizontal"
    android:layout_row="0"
    android:gravity="center"
    android:inputType="number"
    android:text="5"
    android:textAppearance="?android:attr/textAppearanceLarge">

    <requestFocus />
</EditText>
```

Buttons definieren

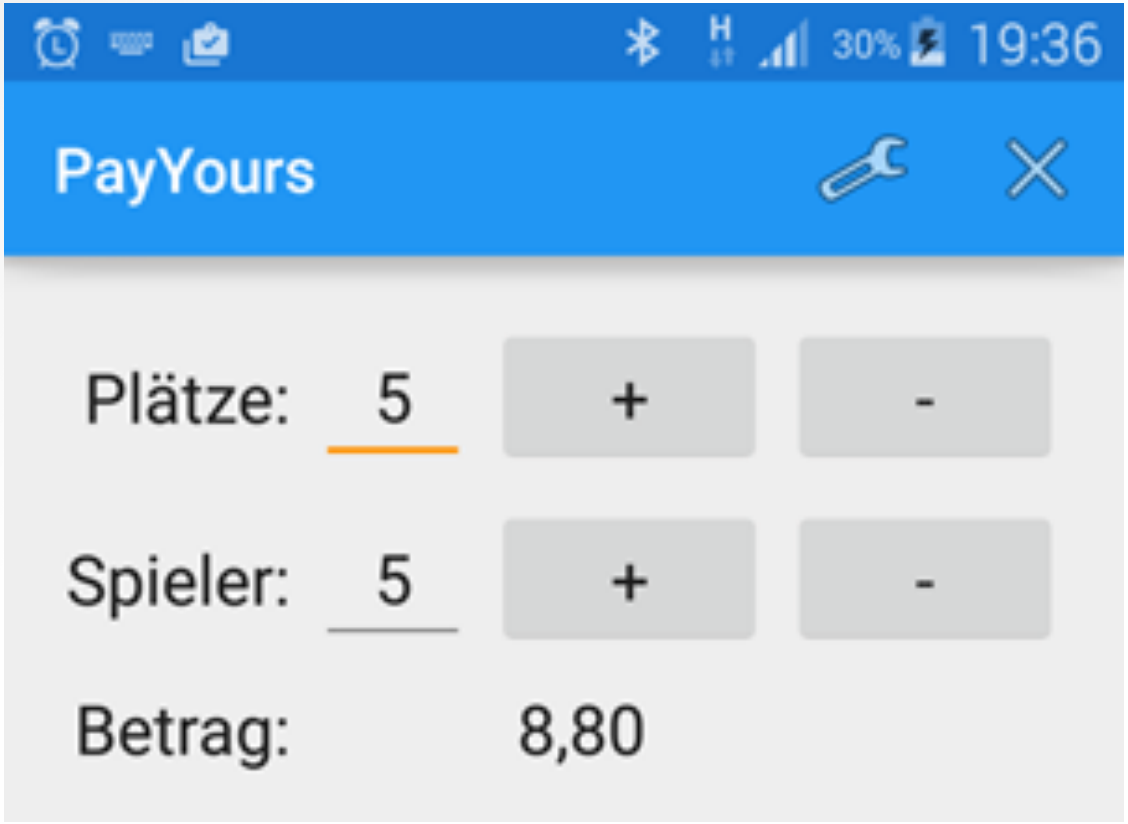

```
<Button
    android:id="@+id/button_plus_courts"
    android:layout_column="2"
    android:layout_gravity="center"
    android:layout_margin="10dp"
    android:layout_row="0"
    android:text="+"
    android:textAppearance="?android:attr/textAppearanceLarge" />
```

Ergebnis über TextView mittig ausgeben




```
<TextView
    android:layout_column="1"
    android:layout_row="2"
    android:layout_columnSpan="3"
    android:id="@+id/textViewTextToPay"
    android:layout_gravity="center"
    android:gravity="fill"
    android:layout_width="110dp"
    android:textAppearance="?android:attr/textAppearanceLarge"/>
```

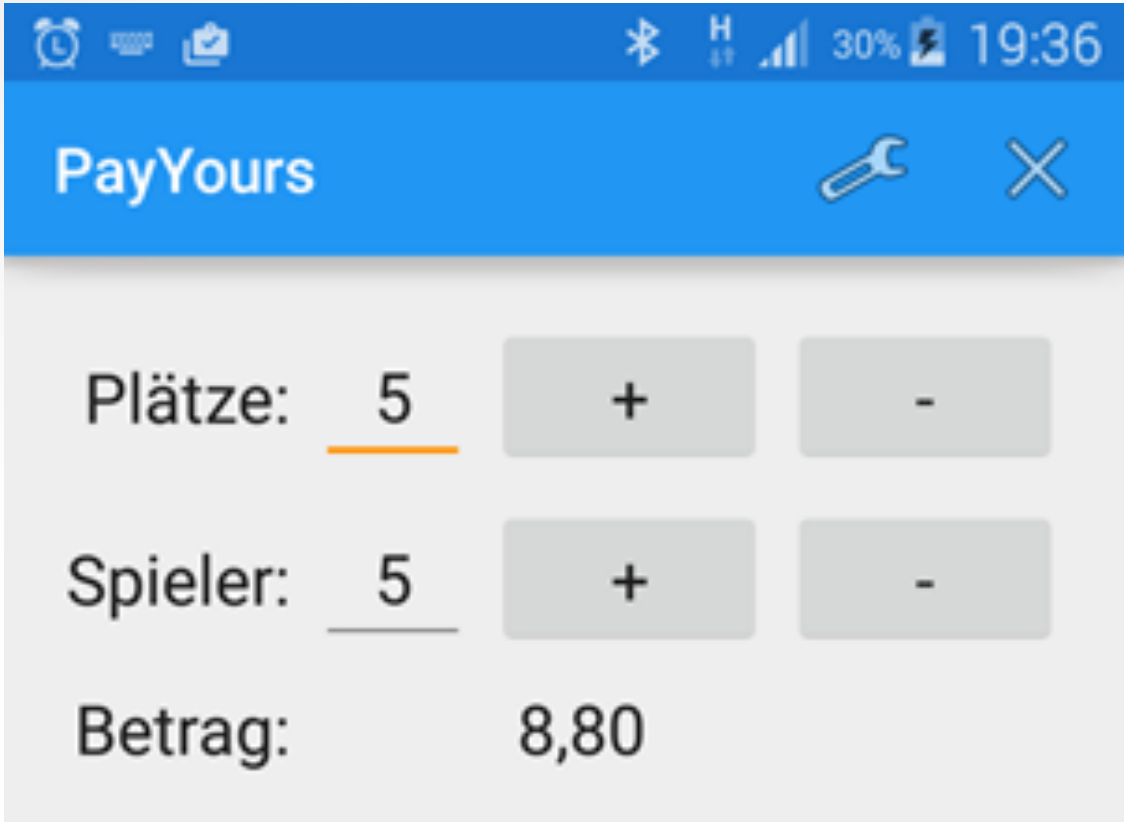
Gewünschtes UI für Calculator



PayYours_b Ende

HTL LEONDING

Grundfunktion implementieren

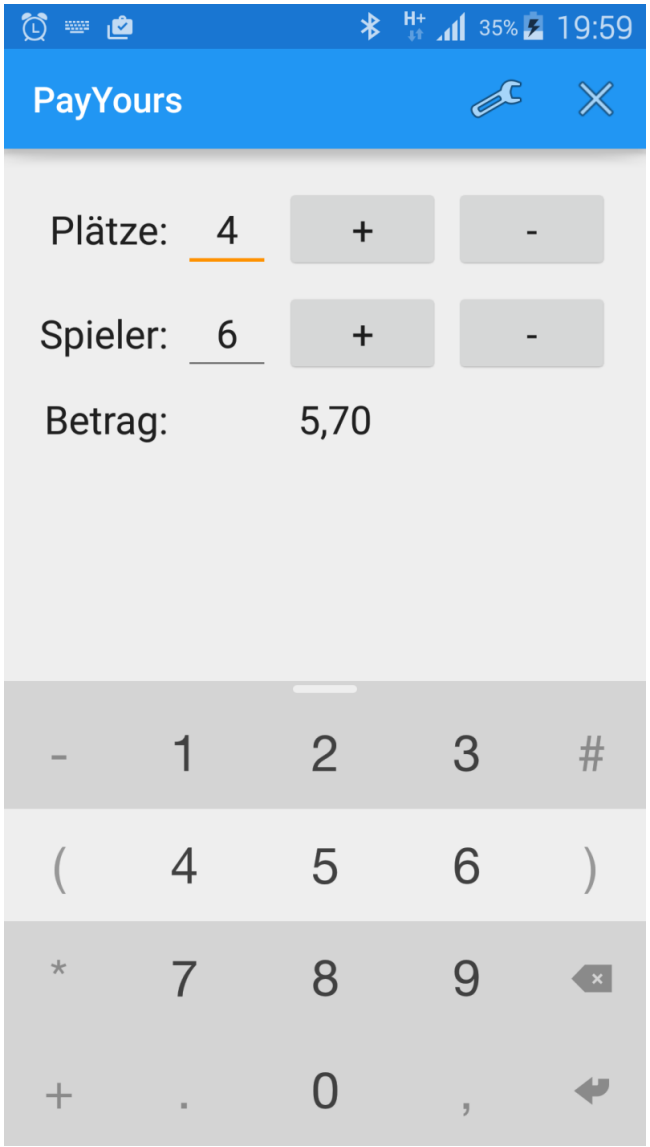


PayYours_c Anfang

HTL LEONDING

MainActivity implementieren (SimpleVersion)

- Steuerung nur über Buttons
  - Keine Eingabe in EditText
- Minus-Button ist ausgeblendet, wenn Zahl <=1 ist
- Ausgangswerte aus Layout
- Preis je Platz als Konstante (8,55 €)
- Ergebnisanzeige nach App-Start und bei jeder Änderung
- Automatische Anzeige des SoftKeyboards mit den Ziffern und dem Komma



HTL LEONDING

MainActivity – onCreateView()

- Layout setzen
- Listener registrieren

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    btn_plus_courts.setOnClickListener(::onClick)
    btn_minus_courts.setOnClickListener(::onClick)
    btn_plus_players.setOnClickListener(::onClick)
    btn_minus_players.setOnClickListener(::onClick)
    et_courts.setOnFocusChangeListener{ view, hasFocus -> onFocusChanged(view as EditText, hasFocus
    et_players.setOnFocusChangeListener{ view, hasFocus -> onFocusChanged(view as EditText, hasFocu
}
```

MainActivity - Manifest

- SoftKeyBoard soll bei Start geöffnet sein ➔ sofortige Eingabe möglich

```
<application
    android:allowBackup="true"
    android:icon="@drawable/payyours"
    android:label="PayYours"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity
        android:name=".MainActivity"
        android:windowSoftInputMode="stateAlwaysVisible">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

MainActivity – onClick-Listener

- Hilfsmethode kapselt Verarbeitung (Codeverdopplung vermeiden)

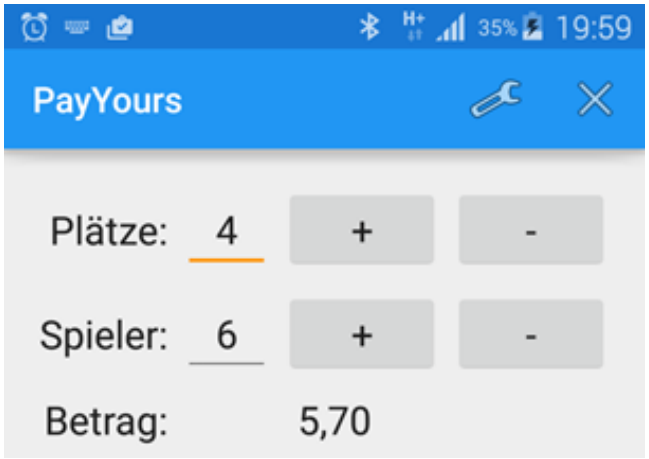
```
private fun onClick(view : View){
    Log.d(LOG_TAG,  msg: "In btnClickHandler for ${view.id}")
    when(view.id){
        R.id.btn_plus_courts -> incOrDecValueInEditText(et_courts,  inc: true)
        R.id.btn_minus_courts -> incOrDecValueInEditText(et_courts,  inc: false)
        R.id.btn_plus_players -> incOrDecValueInEditText(et_players,  inc: true)
        R.id.btn_minus_players -> incOrDecValueInEditText(et_players,  inc: false)
    }
}
```

Hilfsmethode behandelt ButtonClick universell

```
private fun incOrDecValueInEditText(editText: EditText, inc: Boolean) {
    val text = editText.text.toString()
    var number =
    if (text.length == 0) {
        0 // Eingabefeld ist leer
    } else {
        Integer.parseInt(text)
    }
    Log.d(LOG_TAG, msg: "In incOrDecValueinEditText(), number= $number")
    if (inc) {
        number++
    } else {
        if (number > 1) {
            number--
        }
    }
    editText.setText("${number}")
    calculateAndShowResult()
}
```

Main – calculateAndShowResult()

- Der Name ist Programm
  - Werte aus Views auslesen und Ergebnis berechnen
  - Minus-Buttons disablen wenn Wert <= 1 ist
  - Ergebnis ausgeben

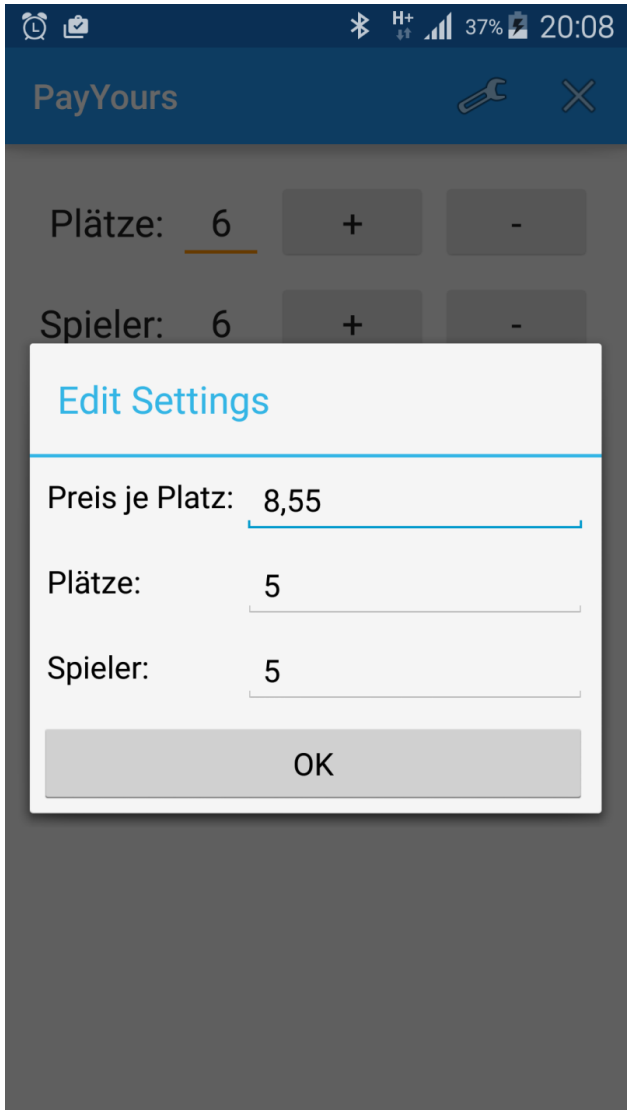


PayYours_c Ende

HTL LEONDING

Einstellungen für Programm speichern

- Grunddaten vorgeben
  - Preis je Einheit
  - Plätze
  - Spieler
- Bearbeitung als Dialog
- Icon für App
- Double im deutschen Format



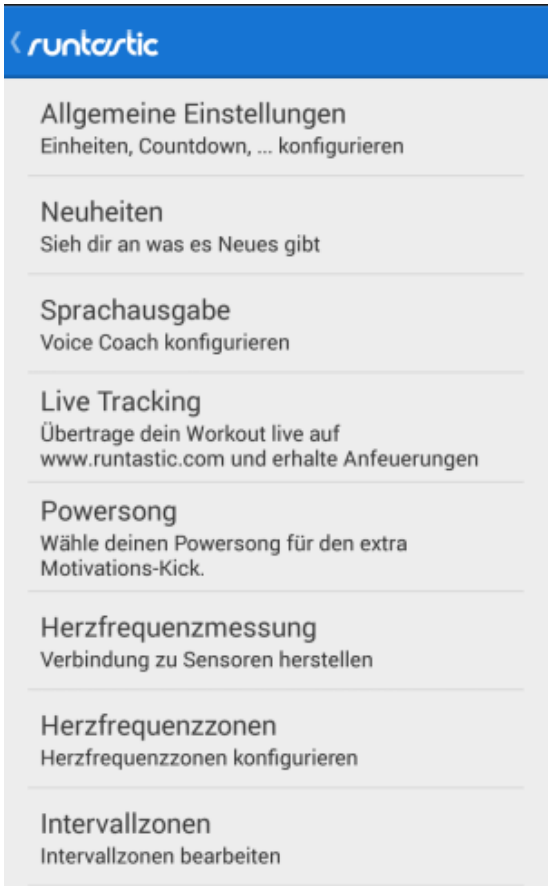
Übung

PayYours_d Anfang

HTL LEONDING

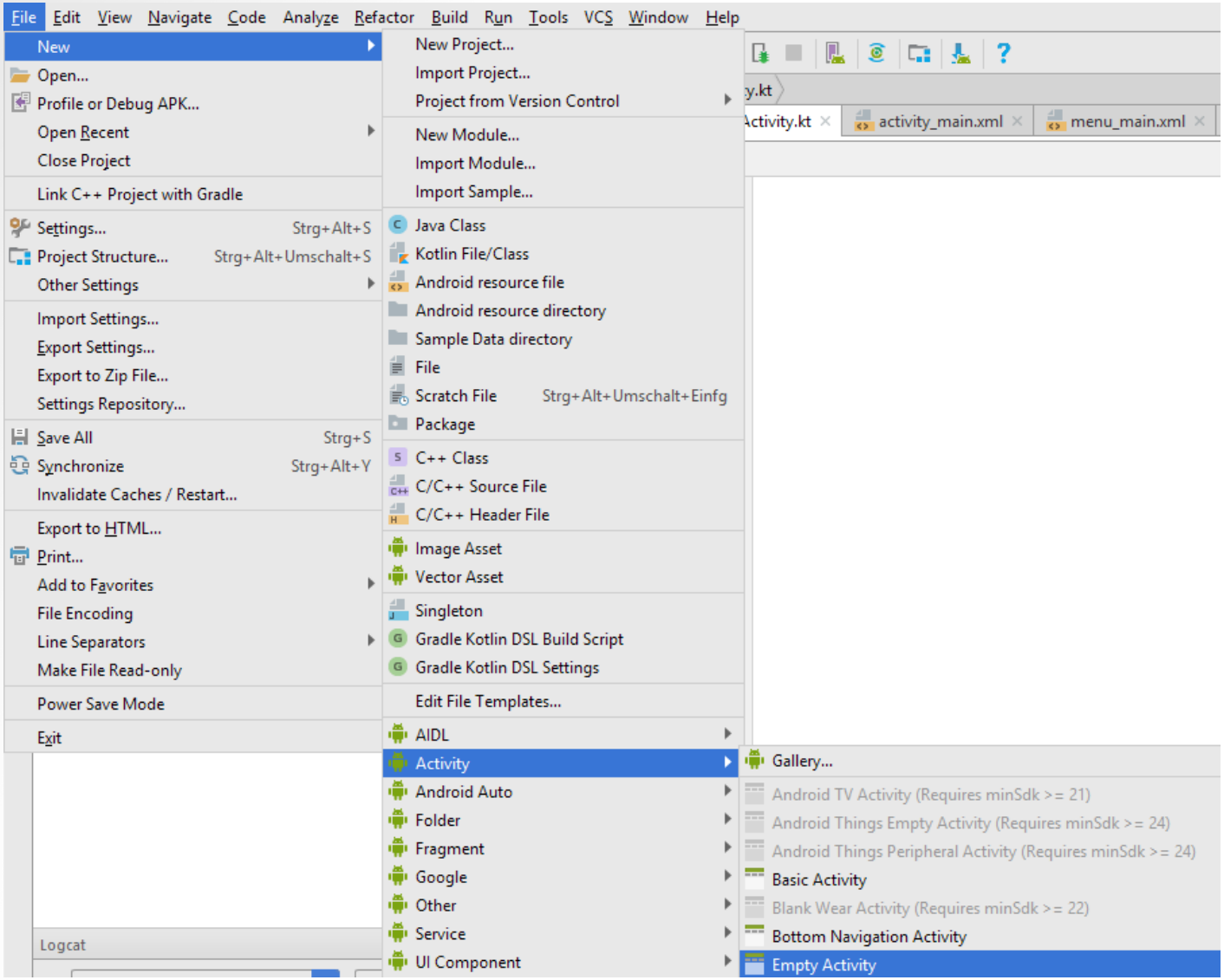
Shared Preferences

- Daten anwendungsspezifisch abspeichern
  - Statusinformationen
  - /data/data/<package name>filename.xml
- Einfache Datentypen
  - boolean, float, int, long, String
- Sehr einfach in der Handhabung
  - Standardmäßig private (Activity oder Application)
- Listener für Änderungen der Preferences möglich
- Preference-Fragments für umfangreiche Einstellungen
  - Untermenüs sind möglich



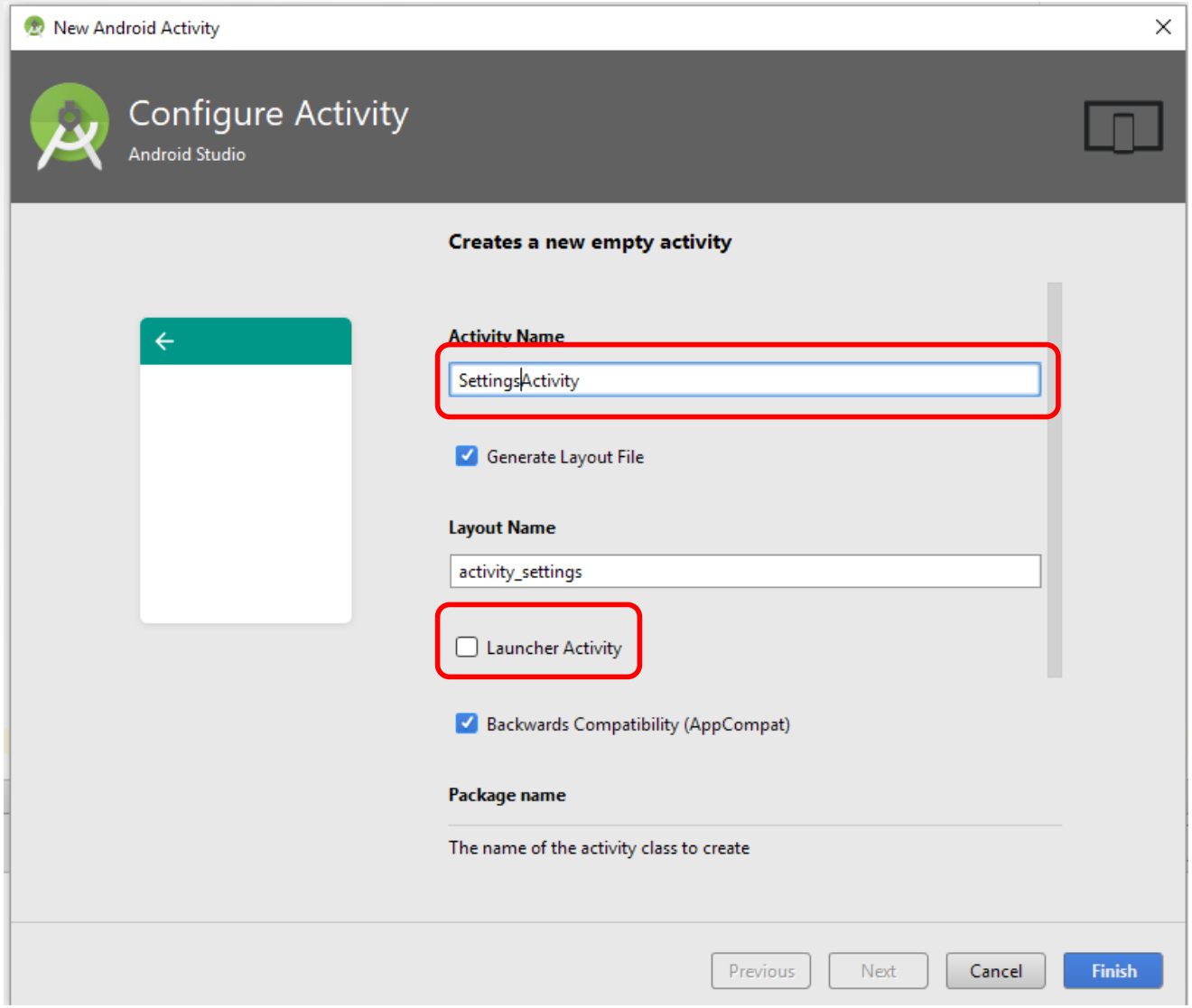
HTL LEONDING

SettingsActivity anlegen



HTL LEONDING

Keine Launcher Activity



HTL LEONDING

Manifest

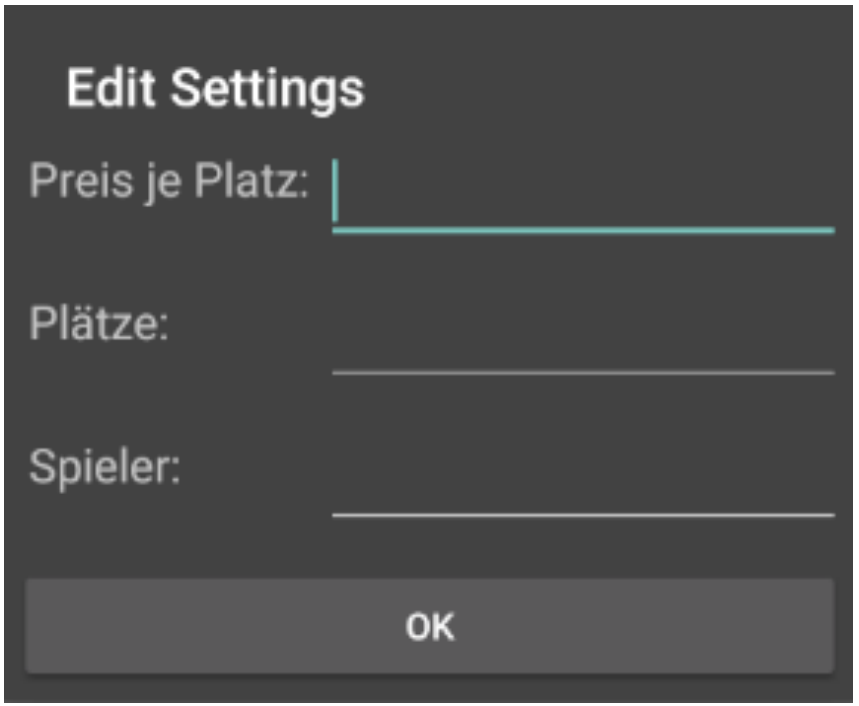
- Dialog-Theme festlegen
- Titel definieren

manifest | application | activity

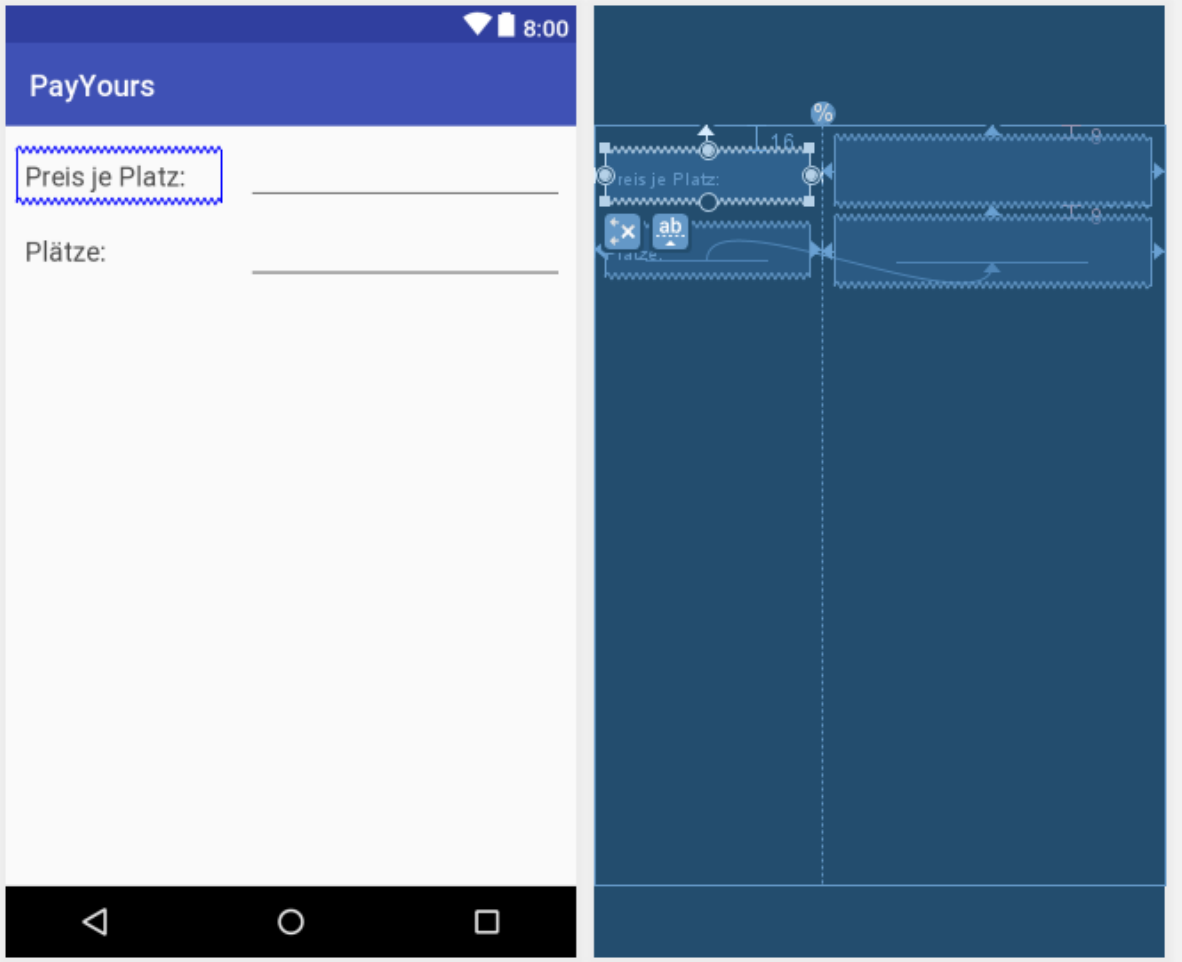
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="at.htl.payyours">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PayYours"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"...>
        <activity android:name=".SettingsActivity"
            android:label="@string/activity_edit_settings_title"
            android:theme="@style/Theme.AppCompat.Dialog"
            ></activity>
    </application>

</manifest>
```

SettingsActivity anlegen

- Layout definieren
  - TableLayout oder ConstraintLayout



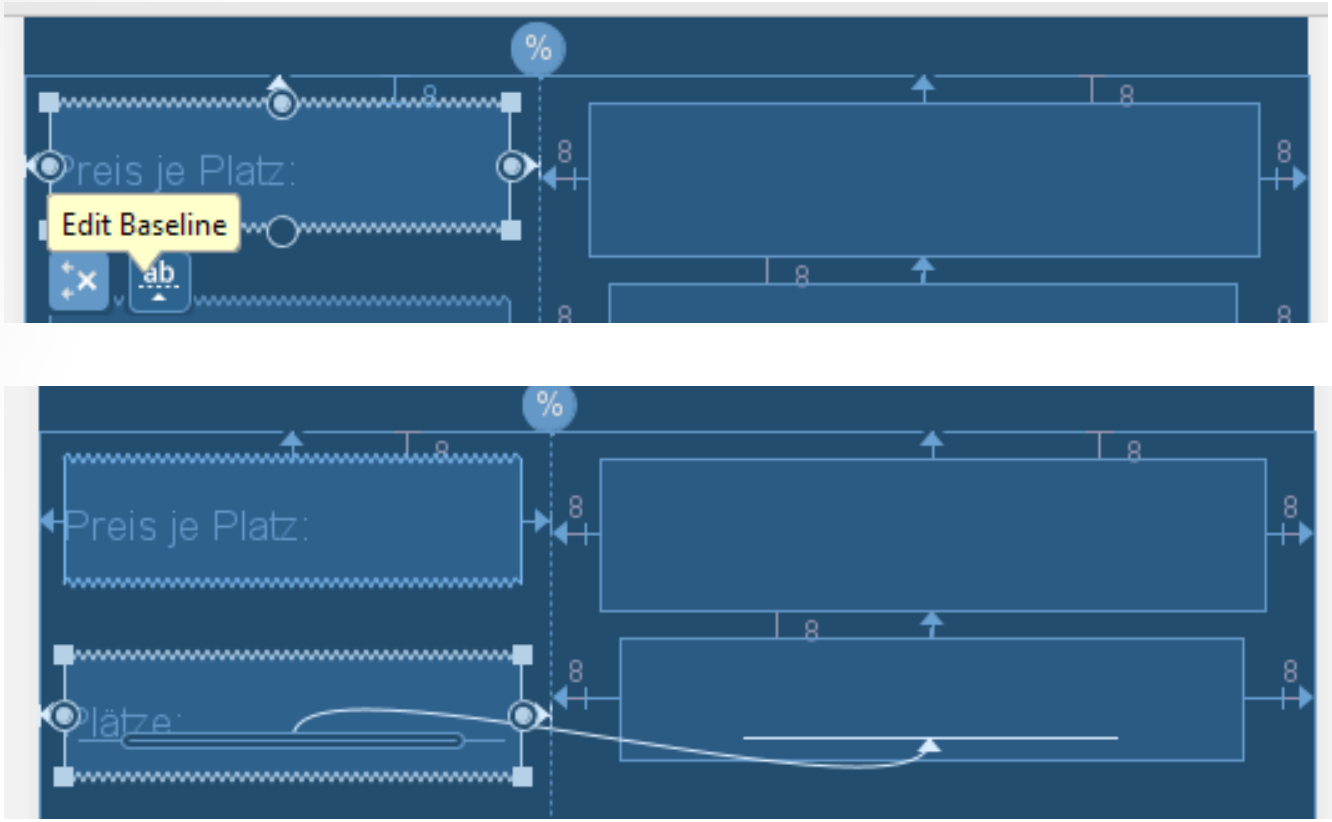
ConstraintLayout definieren



HTL LEONDING

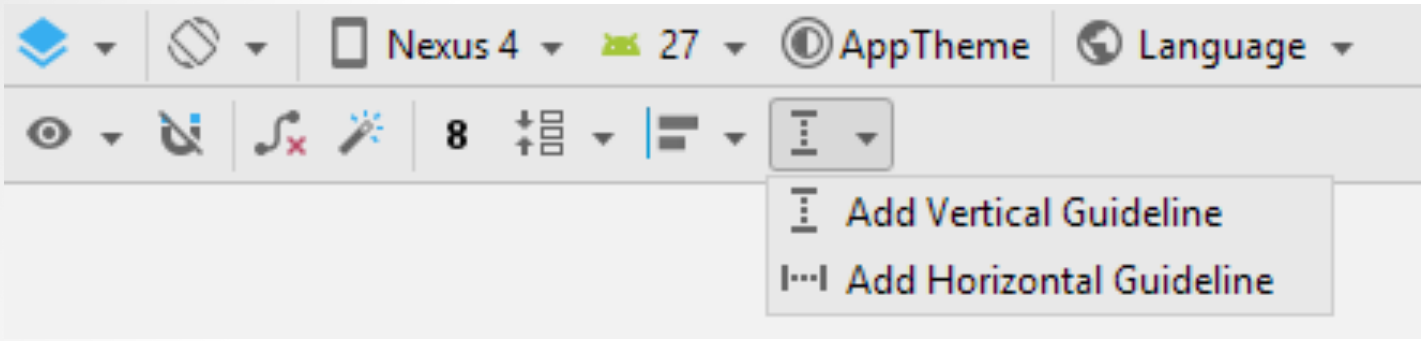
Label und EditText auf gleicher Höhe

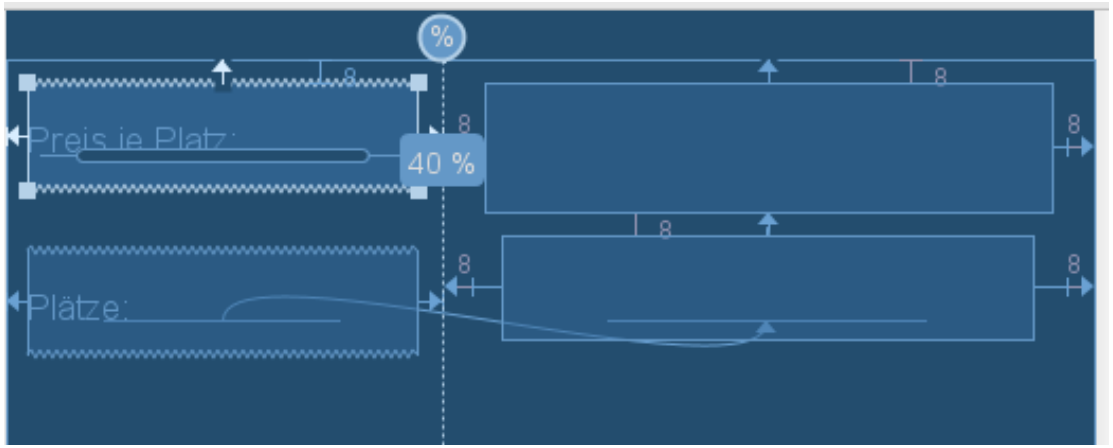
- Baseline konfigurieren



HTL LEONDING

Guideline für saubere Spaltentrennung





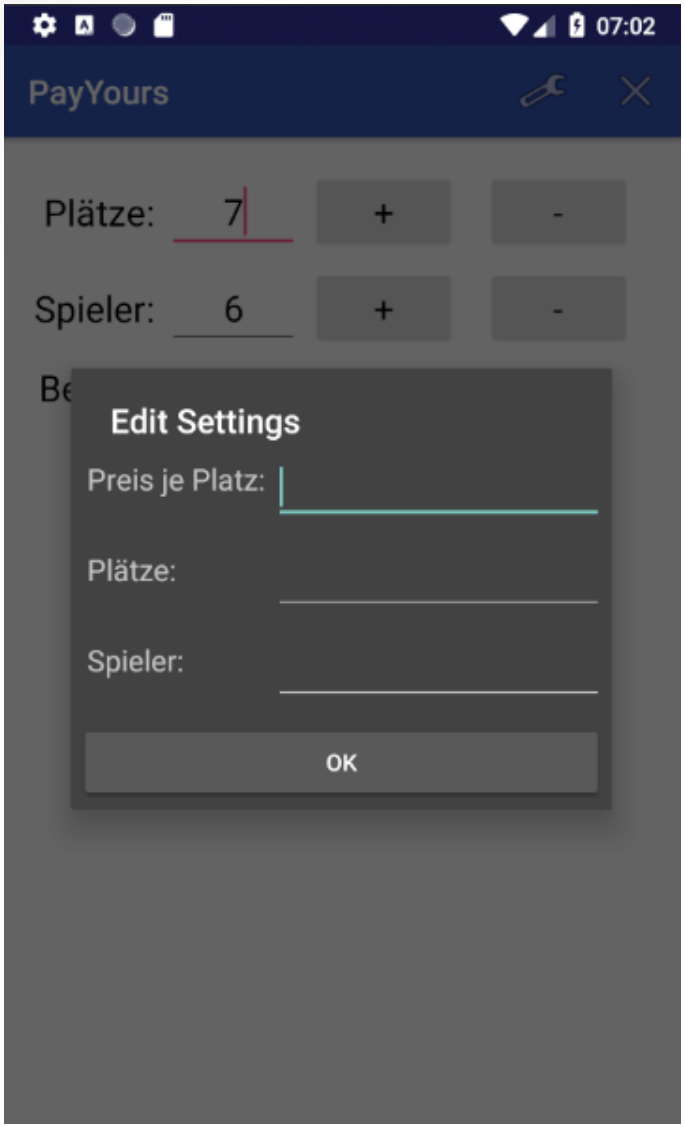
HTL LEONDING

SettingsActivity starten

- Reaktion auf Menüauswahl

```
override fun onOptionsItemSelected(item: MenuItem) = when (item.itemId) {
    R.id.menu_Quit -> {finish();  true}
    R.id.menu_action_settings -> {
                    startActivity(Intent( packageContext: this, SettingsActivity::class.java))
                    true }
    else -> super.onOptionsItemSelected(item)
}}
```

Zwischenergebnis Layout



HTL LEONDING

Settings.kt

- Konstanten für SharedPreferences

```
private const val PREFERENCE_FILENAME = "PayYoursPreferences"
private const val DEFAULT_PRICE_PER_UNIT = "8,55"
private const val DEFAULT_PLAYERS = "5"
private const val DEFAULT_COURTS = "5"
private const val PRICE_PER_UNIT_KEY = "PRICE_PER_UNIT"
private const val PLAYERS_KEY = "PLAYERS"
private const val COURTS_KEY = "COURTS"
```

onCreate() ➔ Werte einlesen

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_settings)
    val preferences = this.applicationContext.getSharedPreferences(
            PREFERENCE_FILENAME, Context.MODE_PRIVATE)
    et_price_per_unit.setText(preferences.getString(PRICE_PER_UNIT_KEY, DEFAULT_PRICE_PER_UNIT))
    et_players.setText(preferences.getString(PLAYERS_KEY, DEFAULT_PLAYERS))
    et_courts.setText(preferences.getString(COURTS_KEY, DEFAULT_COURTS))
}
```

onSave() ➔ Speichern und beenden

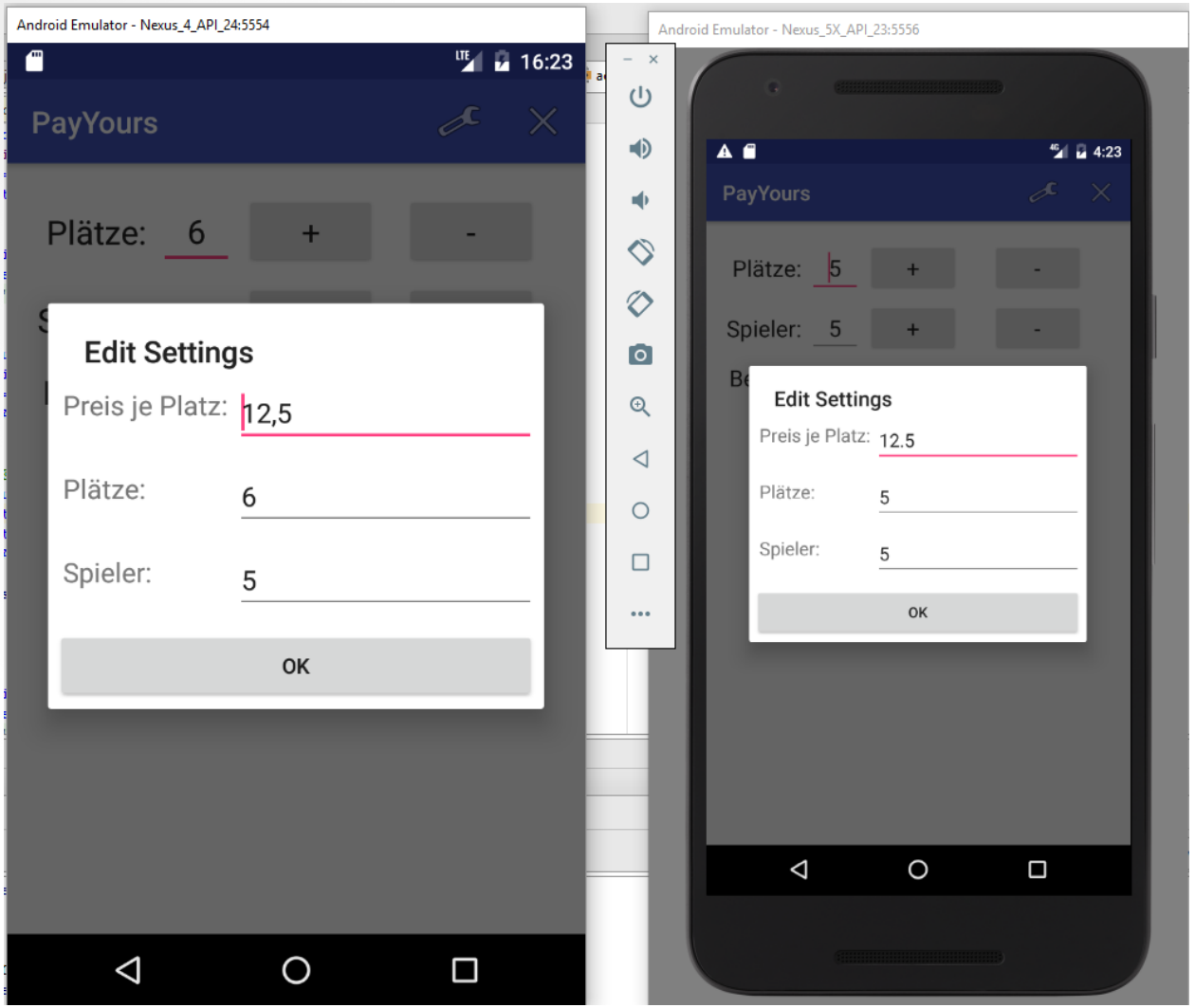
```
private fun onSave(){
    val sharedPreferences = this.applicationContext.getSharedPreferences(
            PREFERENCE_FILENAME, Context.MODE_PRIVATE)
    val preferenceEditor = sharedPreferences.edit()
    preferenceEditor.putString(PRICE_PER_UNIT_KEY, et_price_per_unit_pref.text.toString())
    preferenceEditor.putString(PLAYERS_KEY, et_players_pref.text.toString())
    preferenceEditor.putString(COURTS_KEY, et_courts_pref.text.toString())
    preferenceEditor.commit()
    Log.d(LOG_TAG,  msg: "onSave(), pricePerUnit= ${et_price_per_unit_pref.text.toString()}")
    finish()  // Beendet Activity
}
```

MainActivity ➔ Einlesen

```
override fun onResume() {
    super.onResume()
    readSettings()
}

private fun readSettings() {
    Log.d(LOG_TAG, msg: "readSettings()")
    val preferences = this.applicationContext.getSharedPreferences(
            PREFERENCE_FILENAME, Context.MODE_PRIVATE)
    try {
        val decimalFormat = DecimalFormat.getInstance(Locale.getDefault()) as DecimalFormat
        val numberText = preferences.getString(PRICE_PER_UNIT_KEY, DEFAULT_PRICE_PER_UNIT)
        Log.d(LOG_TAG, msg: "readSettings(), pricePerUnit-Text $numberText")
        pricePerUnit = decimalFormat.parse(numberText).toDouble()

    } catch (e: ParseException) {
        Log.e(LOG_TAG, msg: "save() format exception für float: "
                + preferences.getString(PRICE_PER_UNIT_KEY, DEFAULT_PRICE_PER_UNIT))
    }
    et_players.setText(preferences.getString(PLAYERS_KEY, DEFAULT_PLAYERS))
    et_courts.setText(preferences.getString(COURTS_KEY, DEFAULT_COURTS))
    calculateAndShowResult()
}
```

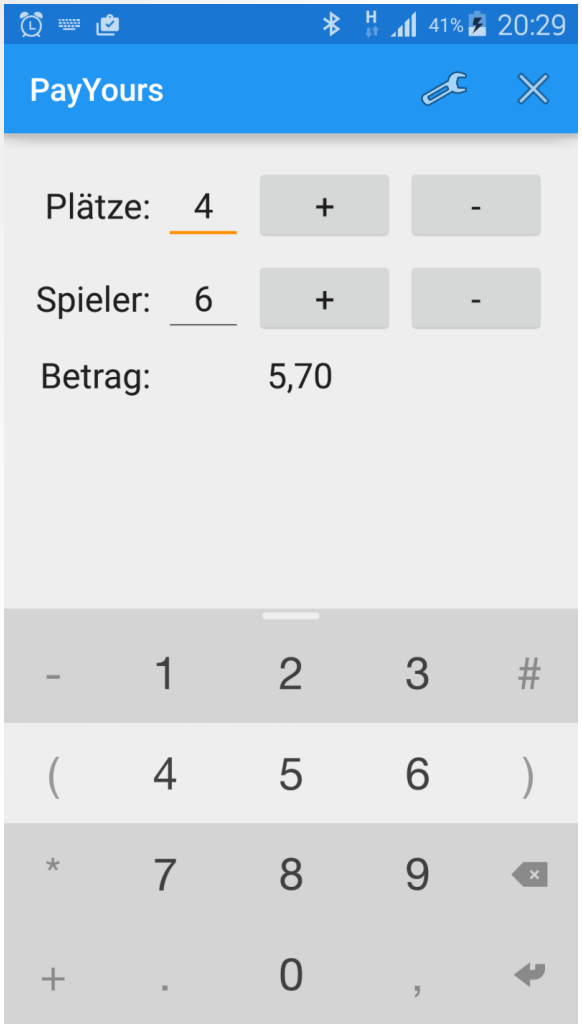
Spezialität – Komma bei Eingabe

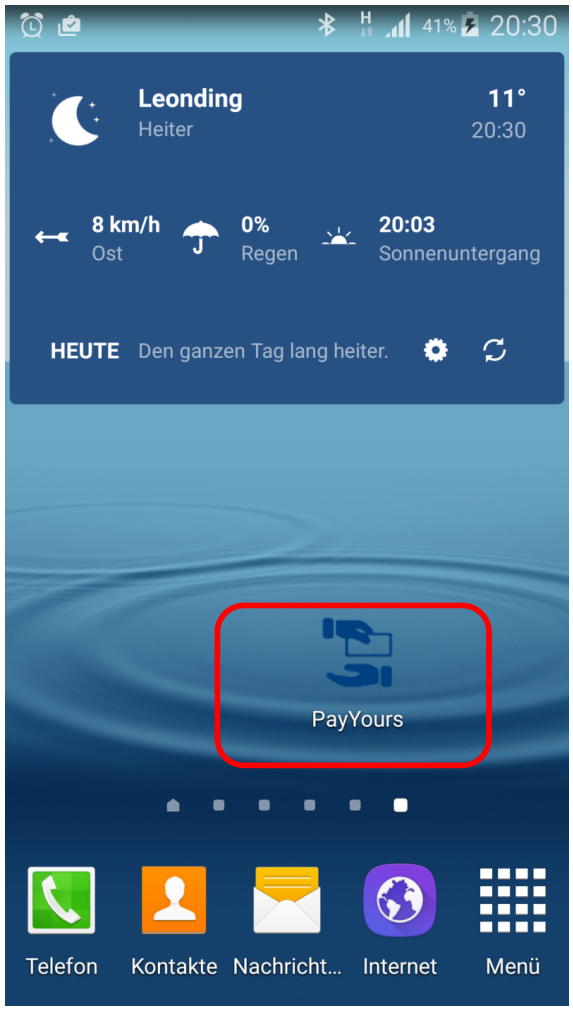
- Tastatur einschränken, richtig parsen und darstellen



HTL LEONDING

PayYours – eigenes Icon





HTL LEONDING

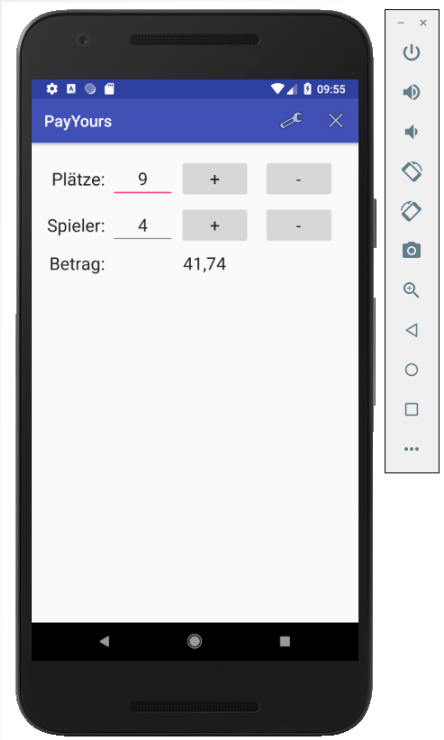
Icon in drawable

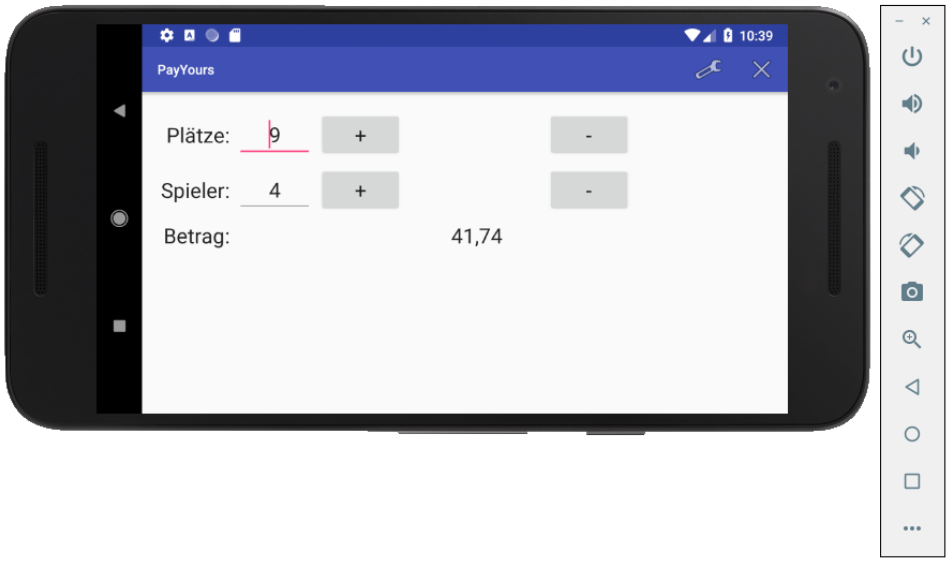
manifest | application

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="at.htl.payyours">

    <application
        android:allowBackup="true"
        android:label="PayYours"
        android:icon="@drawable/payyours"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"...>
        <activity android:name=".SettingsActivity"
            android:label="Edit Settings"
            android:theme="@style/Theme.AppCompat.Dialog"
            ></activity>
    </application>

</manifest>
```

Zustand retten über Orientierungswechsel

- Spezialistenaufgabe
  - Settings müssen auch noch funktionieren





PayYours_d Ende

HTL LEONDING